Glenn McCrea

1. Implicit type conversion in arithmetic operations.

1. Create 2-3 interesting examples of your chosen type conversion in action.
   a. Include at least ONE scenario where type conversion leads to an unexpected result.
2. Provide an explanation for each example, detailing what happens and why.

I tried spending some extra time with boolean conversion because it was somewhat confusing to me. But, I've found it's actually fairly straight forward when you take your time to work through what it's doing.

One scenario that I thought was an interesting and unexpected result was when I wrote:

```
llet alpha = 9999999999999999999;
let bravo = "5";
let SumAlphaBravo = alpha + bravo;
console.log(SumAlphaBravo);
console.log(typeof SumAlphaBravo);
```

And the result from the console was: 10,000,000,000,000,000,005 and that it was a string (but without the commas – I added those now).

What was interesting about this was that I was expecting some sort of an error for such a big number or for it to simply put: "99999999999999999995" but instead, even though the result is a string, it changed it to 10,000,000,000,000,000,005. It simply rounds when it gets to be such a big number and then adds the 5 to the resulting string. It does not do arithmetic.

Then as my second test, I reverse it by subtracting 5:

```
let alpha = 9999999999999999999;
let bravo = "5";
let alphaBravoMath = alpha - bravo;
console.log(alphaBravoMath);
console.log(typeof alphaBravoMath);
```

And the resulting console was: "10000000000000000000 number". Interesting. Subtraction likely gave it an actual number because of the implicit conversion. But then because it was so large, it still rounded up, but kept the data type as a number by converting the "5" to a number.

All in all, it was interesting to see what happened with arithmetic conversion going both ways with a very large number.